Research Article

Automatic Selection and Parameter Configuration of Big Data Software Core Components Based on Retention Pattern

Ping Xu

School of Information Science and Technology, Taishan University, Taian, Shandong 271000, China

Correspondence should be addressed to Ping Xu; taiantsuxp@tsu.edu.cn

Received 13 December 2020; Revised 1 January 2021; Accepted 13 January 2021; Published 22 January 2021

Academic Editor: Yi-Zhang Jiang

Copyright © 2021 Ping Xu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper conducts an in-depth analysis and research on the automatic selection and parameter configuration of the core components of Big Data software by using the retention model and the automatic selection of Big Data components by establishing a standardized requirement index and using the decision tree model to solve the problem of component selection in Big Data application development. By establishing standardized demand indicators and based on the retention model, a data transmission intermediate platform for bidirectional data detection is proposed based on the three demands of user input: storage, computation, and analysis, as well as the problem of undetectable packet loss in data transmission of existing IoT and Web service platforms. The data communication module of the data transmission intermediate platform enables mutual monitoring and detection of data interaction between IoT smart terminals and cloud platforms. The retention mode is built separately to realize the automatic selection of Big Data components. In this paper, we start from several mainstream distributed storage systems and use Cassandra as an example for experiments and tests. We use the multiple regression fitting method to build a corresponding performance model for hardware parameters, take user requirements as input, and use the performance model to configure system hardware parameters; by studying its system principle, architecture, features, and application scenarios, we build a software parameter configuration knowledge base to guide the software. This solves the difficult problem of selecting, deploying, and configuring parameters for Big Data applications.

1. Introduction

Big Data technology is no longer unfamiliar to us, and applications of Big Data technology are everywhere. How to use the value of Big Data more effectively to serve us has become the direction of a malefactor for many people [1]. If we want to take advantage of the value of Big Data, we must process the data. Common Big Data task processing processes include data decompression, data cleansing, data loading, data conversion, and data backup [2]. Scheduling systems take advantage of these interdependencies to schedule these tasks, automatically scheduling each job according to their interdependencies to reduce manual operations [3]. A Big Data application scheduling system can not only realize the scheduling of these simple tasks but also the scheduling management of complex Big Data tasks [4]. These system components with similar functions make it

difficult to select the right system for Big Data application development [5]. Different applications have different emphases. First of all, it is necessary to consider the type of data analysis in the Big Data application, whether it needs real-time processing or batch processing, which directly affects whether to use a Big Data system similar to the Hadoop MapReduce ecosystem or a Big Data system similar to the Spark ecosystem [6]. Furthermore, the method of data processing, such as predictive modelling, ad hoc querying, report generation, and machine learning, directly determines which Big Data tools are used as part of the overall solution [7]. Besides, the frequency and size of data generation, the type of data, the format of the data content, the data source, and the consumers of the data all need to be considered [8].

Azkaban is a batch workflow task scheduling system open-sourced by Wu and is one of the workflow scheduling

systems in the Big Data domain [9]. It is mainly used to run a set of jobs and processes in a specific order within a workflow, which defines a file format for creating dependencies between tasks and provides a Web user interface to maintain and track workflows [10]. Maudsley et al. have designed an indexing analysis utility that performs system performance [11]. Zhang et al. solved the problem of ranking the impact of different parameters on performance by using P&B (Plackett & Burman) statistics [12]. Mohamed et al. designed an automatic performance tuning tool for database configuration parameters, iTunes, which achieves high tuning efficiency, low tuning overhead, and cross-platform portability [13]. iTunes for Oracle database system mainly adopts two approaches: bottleneck elimination and active monitoring [14]. The former focuses on eliminating resource bottlenecks, while the latter uses a comparative approach that periodically monitors the system's runtime data and compares it to historical behavioural data to identify the root cause of performance problems and thus optimize the system's performance [15]. Bendre and Thool studied the performance of distributed storage systems and established a data distribution model for server performance, which effectively reduced the average service time of customer requests [16]. Dishon modelled the performance of distributed file systems and proposed a prediction model that can predict the performance of Lustre and HDFS and improved the HDFS write operation, and the experimental results showed that the model prediction error is within the acceptable range [17]. Based on the runtime characteristics of I/O load, Stilo et al. extracted the performance model of the system online with the help of multiple regression analysis theory [18].

With the development of the mobile Internet, the sources of data are diversifying, the amount of data is exploding, the collection of data is becoming increasingly perfect, and the extraction of useful data from the massive amount of data is becoming increasingly important. As a result, various industries are shifting from data collection to data processing to fully exploit the value of data. As the business of large companies is complex and difficult to process, they are developing scheduling systems that fit their own business needs. To adapt to the characteristics of the autonomous container cloud, this paper designs a logging collection algorithm that adapts to the system load and reduces the impact of logging on the performance of the container cloud platform, thus ensuring the efficient and reliable operation of the services deployed on the autonomous container cloud, as well as guaranteeing a certain comprehensiveness of the log data. To perform the log analysis efficiently, this paper provides a detailed malefactor and product requirements planning for the log data analysis phase, analyses the factors that affect the performance of the sparks, and optimizes the performance of the sparks cluster by taking into account the characteristics of the autonomous container cloud.

1.1. Automatic Selection and Parameter Analysis Design of the Core Building Blocks of the Retained-Mode Big Data Software

1.1.1. Retention Pattern Analysis. The data frame of the communication between the smart terminal and the Web service platform consists of three parts: the header, the tail, and the data recognition area. The frame header represents the start position of a frame; the frame tail represents the end position of a frame; and the data recognition area consists of four main parts: the subsequent data length, the recognition bit, the data area, and the check digit [19]. When the data receiver receives the frame header, the subsequent data received are the effective data length, and then, the data receiver receives the data corresponding to the effective length according to the length, and finally, the complete reception of the frame is realized. The identification bit in the data recognition area is used to indicate the identification mark of the data, indicating which data type the data belong to. The check digit in the data recognition area is used to accumulate the data in the data area, and then, the accumulation result is used to perform a remainder operation on 0xFF to achieve the checking purpose. The main function of the data area is to store valid data, and it is possible to determine the format differences of the data frames in this area. Generally, one of the data frames is sent to the Web service platform via the intelligent terminal, and the other is sent to the intelligent terminal via the Web service platform, as shown in Figure 1.

The goal of this research is to design and implement an autonomous container cloud big user data log analysis platform for the distributed collection, storage, and processing of the collected logs using real-time and offline Big Data processing techniques and finally to process the processed logs. The results and raw log data are visualized and presented to the users of the autonomous container cloud as well as the development and operation personnel. The distributed log data storage module provides support functions for the Big Data analysis system of the logs of the entire autonomous container cloud platform, in which four functions are required; the first is the log data storage function, which is the most basic function of the module, where it is not only necessary to store the raw log data collected and preprocessed by the distributed log collection system [20]. It is not only necessary to store the raw log data collected after the collection and preprocessing of the distributed log collection system but also to save the results calculated and analysed by the Big Data computing module. Especially for the raw logs, the amount of data is very large; therefore, the distributed log data storage module needs to have good access performance and large storage capacity. Secondly, to facilitate developers to query the raw logs and result logs, the distributed log data storage module needs to have an efficient query function and convenient log query interface, which is also convenient for Big Data Log Analysis

FIGURE 1: Schematic representation of the retention pattern.

Module to obtain logs from the distributed log storage module. The third is to provide simple statistical analysis functions for data logs, as well as certain search functions and log data aggregation functions [21]. The distributed log data storage module requires simple statistics and filtering functions for log information, which not only facilitates developers and operation and maintenance personnel to retrieve and view the original logs but also improves the efficiency of the Big Data Log Analysis Module in extracting the original log information from them. The distributed log data storage module is not isolated but needs to cooperate with other modules, which requires this module to have a perfect open interface to achieve the coordination and unification of the whole Big Data analysis system. Generally, the messaging middleware is mainly used to forward application events and data to the specified service program, and then, the service program processes the data through the business logic, and the role of message middleware in a distributed system is shown in Figure 2 when the system uses message middleware; the messaging middleware mainly consists of producer, consumer, and broker. The consumer is the role used by the client and the broker is the equivalent of a server to relay messages from the client.

In this paper, we take Cassandra's write throughput performance modelling as an example for detailed analysis and use a similar approach for the remaining four performance metrics. Ignoring other minor factors, consider a server with memory size M (in GB) and CPU core count C . The Cassandra write throughput (in B/s) deployed on the server is assumed to be

$$T_w(C, M) = f(C_w, M_w). \quad (1)$$

Because Cassandra is a distributed, P2P architecture, highly scalable data storage system, it has good horizontal scalability. Based on the horizontal scalability, the write throughput of a Cassandra cluster with N machines is as follows:

$$\begin{aligned} T_w(N, C, M) &= f(C_w, M_w) \times f(N_w, C_w, M_w), \\ f(C_w, M_w) &= a_1 + a_2 C_w + a_3 N_w + a_4 M_w + a_5 C_w N_w M_w, \\ f(N_w, C_w, M_w) &= a_1 + a_2 C_w \times N_w + a_3 N_w \times M_w \\ &\quad + a_4 M_w \times C_w + a_5 C_w N_w M_w. \end{aligned} \quad (2)$$

Batch processing files are an extension of scripts, which are closer to natural language, which is an advantage for normal application development. The only prerequisite for cross-platform execution is that the corresponding language interpreter is available on the corresponding system. Therefore, scripts are characterized by invoking system-related commands to perform many related tasks, rather than by forcing a series of tasks to be performed automatically. Scripts are combinations of macro commands, as are system initialization, service configuration, compiled configuration, automated testing, and batch processing.

$$\text{tab}_R(i) = r((G_{\text{mean}} + 1 + M - G_{\text{mean}-2}) \times c d f_R(i)),$$

$$D_R(i) = \frac{h(i)}{N_R}, i \in [0, M-1], \quad (3)$$

$$\begin{bmatrix} B - MC^{-1}M^T & 0 \\ M^T & C \end{bmatrix} \cdot \begin{bmatrix} \Delta x_C \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} v - MC^{-1} \\ w \end{bmatrix}.$$

1.1.2. Automatic Big Data Software Core Component Selection and Parameter Configuration Analysis. In DAG workflow-type scheduling systems, the flexibility of task triggering mechanisms and the complexity of job dependencies make the functionality provided more complex and the problem more difficult to solve. Task prioritization, service isolation, and permission management are common in such scheduling systems. In this case, service isolation at an executing end becomes unavoidable and only the nodes

FIGURE 2: TSM file structure.

registered in a particular service perform a particular task [22]. Also, service links are generally short and real-time requirements are high, so priority management is low. Resource isolation is relied upon for resource availability, and there is little possibility of competing for resources, as is rights management.

Also, it is very common for many jobs to be executed with shared resources, thus highlighting the issues of prioritization, load segregation, and rights management. Processes such as pausing tasks, reflashing history, and manually marking failures or successes are essentially caused by the complexity of the business process, which requires flexibility and robustness, as well as complete monitoring and alarm notification mechanisms, such as simple task failure alarms, timeout alarms, traffic load monitoring, and business progress monitoring and prediction. Also, it should have complete monitoring and alarm notification mechanisms, such as simple task failure alarm, timeout alarm, traffic load monitoring, and service progress monitoring and prediction. After analysing and investigating the scheduling system, I finally decided to develop a workflow scheduling system suitable for my company's business scenario based on the open-source DAG workflow scheduling system. As an open-source workflow scheduling system, these open-source scheduling systems are currently used more in the industry. The analogy with Hadoop integration and maintenance costs was investigated in several aspects, as shown in Table 1.

Execution logic is at the heart of system scheduling. Tasks that are solved with Big Data will become more difficult. A complete business workflow usually requires multiple subtasks to work together. There are strict dependencies between subtasks. At the same time, to increase the concurrency of the system, there is a parallel relationship between various task workflows, while other tasks have

many different processes. When multiple tasks need to be handled simultaneously, a unified task scheduler is needed to coordinate them so that they can be executed smoothly and server resources can be controlled and utilized efficiently. Over time, information systems in various companies have accumulated large amounts of data. In the past, only one small machine was used to process all the data, and the processing results were far from satisfactory. Moreover, the amount of data that could be processed on a small machine was not only limited but also very expensive, with a very low unit input/output ratio. Under such circumstances, a Big Data platform emerged that builds a distributed cluster by aggregating several public servers [23]. It handles large amounts of data with high processing speed, thus achieving high-cost performance, and has been welcomed by many enterprises as a solution for most enterprises to build a Big Data platform. In this context, there is an inevitable trend to develop general Big Data application scheduling systems that support Big Data platforms, including HDFS distributed file system, HBase distributed database, Hive data warehouse tools, and MapReduce computing framework, and Mahout Big Data era data mining and data analysis tools, as shown in Figure 3.

As an important part of the Big Data application scheduling system, the main role of the task dispatcher function is to achieve high availability of the system. To ensure the high availability of the scheduling system, the dispatching function is extracted from the native scheduling system as an independent dispatching subsystem. The dispatching functions of the native system are integrated into the execution machine, which integrates dispatching and execution. This design is fine and lighter when the volume of business is small, but when the business becomes more complex, various problems arise, such as the frequent

TABLE 1: Scheduling system analogy table.

Project name	Oozie	Azkaban	Airflow
High availability	High	High	High
Ease of use	General	Better	Better
Community activity	In	In	In
Scalability	General	High	General
Integration with Hadoop	Go deep	General	General
Maintenance cost	High	In	In

FIGURE 3: Big Data software core components automatic selection and parameter configuration framework.

accumulation of execution tasks and distribution tasks that cannot be properly controlled. Therefore, with the help of Zookeeper dispatch function, the problem of complex tasks can be solved with high concurrency and the point of view of software engineering is reduced. Through the Zookeeper leader election, the distribution node in the event of failure is automatically switched. Through the Zookeeper leader election, the distribution node in the event of failure to automatically switch. This not only can quickly expand the horizontal capacity of the task execution nodes but also convenient in the management side of these nodes to control.

Security has always been an important part of any system that cannot be ignored. As a dispatch system, security is inevitably something that cannot be neglected. The security of a scheduling system is mainly manifested in two aspects: project management authority validation and data security. Project management security mainly requires permission control of projects, where each developer only has the

permission to manage the projects he creates while the administrator can manage all projects. Data security mainly involves taking certain security measures for the database to prevent data loss in case of a database failure, backing up the necessary data, and also making sure that the core code of the system cannot be leaked.

1.1.3. Validation Design Analysis. As a Big Data application, the performance requirements of the scheduling system are also a key factor affecting scheduling usage. Since the scheduling system may carry many highly concurrent tasks, the performance of the system must be efficient. Moreover, a Big Data application scheduling system acts on a Big Data platform to handle a small number of complex tasks, and the scheduling performance needs to exceed 100,000 levels per hour at least [23]. Only when such performance requirements can be met, the scheduling system can work properly. In practice, the normal operation of tasks depends largely on

the stability of the system. Moreover, in many cases, different kinds of workflows are scheduled together, and there are complex dependencies between them, which requires that an abnormality in the scheduling of anyone's workflow cannot affect the normal execution of the others. The main purpose is to ensure the high availability of the scheduling system and to extract the dispatch function as an independent dispatch subsystem. Through the zookeeper leader election to achieve automatic switching when the distribution node failure, you can quickly expand the horizontal capacity of the task execution nodes, and the management side can easily control these nodes. The approximate flow of the dispatch module is as follows: the use of zookeeper leader election to select the leader dispatch node; dispatcher from the database periodically to get the flow to be dispatched; dispatcher from the executor to get the current execution of the executor load information; dispatcher according to the load strategy selected by the management side and the executor's resource use to get execute the executor of the flow instance; and push the dispatch results to each executor node, as shown in Figure 4.

If the server has enabled HTTPS configuration, it will find the corresponding digital certificate, which is divided into a public key and a private key. The public key can be made public and is mainly used to encrypt the data, while the private key is used to decrypt the data. After receiving the public key data, the client will perform relevant checks on the data, such as checks on the issuing authority and expiration time. If an exception occurs during the checks, a warning box will pop up to inform the client that there is a problem with the certificate. The server receives the encrypted data, decrypts it with the private key, takes out the random value, and then symmetrically encrypts the previously requested content with the random value. Symmetric encryption is hybrid encryption of the information and the random value through an algorithm so that only the random value can be decrypted if it is known and returns the symmetrically encrypted data to the client.

The parameters collected by the system are mainly the physical data of the industrial workshop, such as the physical parameters of the machine, the inspection data of the products, and the collection data of the electrical energy. The visualization of these data mainly uses curves, charts, and texts for processing. Because the amount of data is too large and the display frequency is too fast, this will result in a large occupation of system resources, or the text data will be too large and quickly updated, which will make it impossible for the naked eye to distinguish the data. When the underlying data are sent to the server, the server stores the data in Influx DB and samples the Influx DB data with its sampler and then sends the data to the Influx DB via middleware or other forms. To improve efficiency, a data cache layer can be added to cache the last hour's data, and the sampler can sample the data directly from the cache so that you do not have to sample the data directly from Influx DB every time [24].

When the server-side data sampling and pushing are complete, the second step of visualization is to design a graphical framework that can respond to the real-time data and automatically draw curves, histograms, and lists based

on the pushed data [25]. Real-time data visualization requires the visualization program to refresh the related visualization curves, text, and charts with the update of the collected data, while historical data visualization requires the visualization program to have a higher degree of freedom to show the collected historical data from multiple dimensions, so their focus and difficulties are not the same, so the system adopts two sets of visualization solutions to solve the real-time data visualization problems, respectively.

2. Results and Analysis

2.1. Analysis of Automatic Selection and Parameter Configuration Results. The experiment tests the maximum performance that can be achieved by continuously increasing the number of client threads. The test uses 10 columns per row, with an average of 10 characters per column. In a multinode environment, a single-copy strategy is used to test the maximum performance it can scale. On this basis, the impact of multiple copies on performance is considered. In this paper, we take Cassandra's write throughput performance as an example, as shown in Figure 5; the horizontal coordinate is the number of nodes and the vertical coordinate is the throughput, and there are three sets of results, namely, 1c2g (1 core CPU, 2 GB of memory, and the same below), 2c4g, and 4c8g. The experimental results show that when the number of CPU cores and memory size are fixed, the write throughput increases with the number of CPU cores and memory size. The number of nodes increases as the number of nodes increases.

The main function of the system's hardware is to collect the manufacturing data, personnel information, and material data of the cable workshop and then transmit these data to the system's software level for related processing and display. The hardware of the system mainly contains the communication network of the system, equipment data collector, electronic label reader/writer for tracking personnel, material, semifinished product information, and some auxiliary sensing devices. This system is to install and deploy some collectors, sensors, communication networks, and other equipment in the workshop to complete the collection and transmission of workshop data, due to the uniqueness of the production workshop, so the collectors, communication devices, and other equipment must ensure their stability so that they can continue to work in a harsh environment without accidents, so the market's civilian routers, switches, and other equipment are not so. The collectors, routers, and switches are all industrial grade, and the collectors and switches are customized according to the field needs. The real-time log data processing module adopts Spark streaming to analyse the log data by performing operations such as a map, reduce, aggregately, and partitions. Data correlations and valuable analysis results that exist within the massive data are sent to a distributed log data storage module, as shown in Figure 6.

When you click Execute Now, you will enter the workflow view interface to configure the workflow in detail. You can configure the actions to be performed when a workflow first appears to succeed or fail, which can be

FIGURE 4: Big Data platform analytics validation flowchart.

FIGURE 5: Cassandra write throughput experimental results with fixed CPU core count and memory size.

divided into failure or success prompts, mainly e-mail or SMS prompts. Then, you can select the operation to be executed when a workflow fails to execute, that is, you can select to complete the currently running job. That is, you can choose to complete the currently running jobs or cancel all the jobs and set the workflow to a failed state or run all the dependencies of the current jobs. Concurrency can also be set, i.e., if the current workflow is running, you can set it to skip execution, run in parallel, or block operations. Finally, you can set the workflow parameters, i.e., you can set the base time and job priority for the workflow execution, as shown in Figure 7.

System testing is also an important part of software engineering. Only with good testing we can ensure the quality of the product and deliver a satisfactory product to

the user. In the previous sections of this paper, we have completed the development and design of the system from the research background and significance, related theories and key technologies, requirements analysis, outline design, and detailed design and implementation of the system. The next step is to verify and test the results of the previous design and development, that is, to test the system. The system testing is done to find as many defects as possible so that the system can be repaired. The task dispatcher is started by the task dispatcher module and runs simultaneously on the Web management configuration. The task dispatcher first needs to be deployed in a cluster, at least two dispatch instances need to be started, and then according to the zookeeper leader election to elect a leader for the task dispatching. Different instances determined whether the

FIGURE 6: Time distribution graph.

FIGURE 7: Write throughput results.

distribution node leader election is carried out properly or not. Then, test the project management through the management side of each workflow setting to dispatch settings which, in turn, test the selection of different load policies to determine whether the distribution of the task with the choice of the distribution strategy. Then, test if each flow instance can only run 5 at the same time by default, and each execution node can run up to 30 flow instances at the same time.

2.2. System Performance Result Analysis. The software is written in WPF + WebView mixed mode, and its main interface is shown in Figure 8, which mainly displays the recent 24-hour data of the “ED” machine and can be used to manage the data of different periods. When the mouse moves to a certain point in time, all the acquisition parameters of that point in time will be displayed. If the frequency is exceeded, the management software will crash because the data refresh frequency is too fast, and the higher

frequency of real-time data can only be reflected in the electronic signage and HMI software.

This interface is mainly used to upload some process parameters and process files, for example, some process parameters in the main interface of the electronic signage in the previous section are maintained by process management. The “Scan Bar Code” option is to bind process parameters to a bar code, and then, you need to use a bar code gun to scan the bar code, as shown in Figure 8. The bar code data can be entered manually and the binding can be done as well. GRDT and IDT algorithms have several advantages regarding algorithm efficiency, privacy protection strength, data availability, and the ability to use the bar code. A pair of experimental results for the four aspects of scalability is shown in Figure 9.

As shown in Figure 9, the execution time of GRDT is shorter than that of IDT for different f -parameters, and GRDT can significantly improve the efficiency of the algorithm by pruning the generalized nodes. As the f -parameter increases, the efficiency advantage of GRDT over DDT

FIGURE 8: System management interface.

FIGURE 9: Running time of the two algorithms.

increases. This is because, as the parameter increases, the total number of generalized lattice pairs satisfying the T -closeness constraint also increases, and the total number of pairs of generalized nodes to be traversed in each round of iteration also increases. The growth test results show that the acceleration ratio of the GRDT algorithm increases approximately linearly as the number of compute nodes increases. GRDT has good scalability, while IDT does not maintain a linear relationship. In contrast, the IDT algorithm does not maintain a linear acceleration ratio, and as the dataset size increases exponentially, the IDT algorithm requires much more than the corresponding multiplier to complete the desensitization in the scale-increasing experiments. This is because, without generalized node pruning, the computational overhead of the IDT algorithm increases rapidly with the number of iterations, which affects the scalability of the algorithm. The scalability of the GRDT algorithm is better than that of the IDT algorithm, as shown

by the computational results of the acceleration ratio and scale growth.

In the workflow execution, you can select the dispatching strategy and set the dispatching instance through the workflow setting function (Figure 10). The workflow configuration page is mainly divided into five options to configure the workflow: dispatch configuration options, execution configuration options, notification configuration options, resource configuration options, and custom properties. Each option can be expanded to configure detailed parameters. When the workflow is executed, the workflow can be configured through the execution settings, notification settings, and resource settings. The existing online Big Data platform technologies for IoT are not perfect in terms of platform performance and user experience, and there is no open-source, easy-to-use private platform technology available on the market, and the service platform is expensive to purchase, and there is no open-source

FIGURE 10: Statistics of analysis results.

technology framework for research and small businesses to use directly. In this paper, we design a Big Data platform for the Internet of Things (IoT) applications, which is mainly divided into two modules: a middle-tier platform for data transmission and processing and a Web service platform for IoT. The Web service platform uses the following modules. The new B/S architecture is a newly emerging Web network architecture pattern that unifies the client.

3. Conclusions

Big Data application systems include data collection, storage, analysis, mining, visualization, and other technical aspects, and there are multiple solutions for each aspect, involving hundreds of different systems and complex system configurations, which brings great challenges for enterprises to build Big Data application systems. According to the user's needs for Big Data storage, calculation, and analysis, the appropriate Big Data storage system, Big Data calculation system, and Big Data analysis system are automatically selected. If the user selects the storage requirement, the corresponding storage system is output; if it is the computation requirement, the storage system is also required. For storage and compute components, the selection of storage and computer systems should be output at the same time; similarly, if the user selects analysis requirements, the selection of storage, compute, and analysis systems should be output at the same time. The algorithm gives a pseudocode for the component selection process. To solve the problem of component selection in Big Data application system development, the automatic selection of Big Data components is realized by establishing standardized demand indicators and adopting a decision tree model. Based on several mainstream distributed storage systems, Cassandra, for example, uses multiple regression fitting to build a corresponding performance model for hardware parameters,

takes user requirements as input, and uses the performance model to configure system hardware parameters; by studying system principles, architecture, characteristics, and application scenarios, it builds a software parameter configuration knowledge base to guide the configuration of software parameters, thus solving the problem of the hardware parameters. Automated component selection and parameter configuration issues in bigdata system are developed.

Data Availability

Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

Consent

Informed consent was obtained from all individual participants included in the study references.

Conflicts of Interest

The author declares that there are no conflicts of interest.

References

- [1] T. Wang, H. Ke, X. Zheng et al., "Big data cleaning based on mobile edge computing in industrial sensor-cloud," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1321–1329, 2019.
- [2] A. Wahyudi, G. Kuk, and M. Janssen, "A process pattern model for tackling and improving big data quality," *Information Systems Frontiers*, vol. 20, no. 3, pp. 457–469, 2018.
- [3] A. C. Belkina, C. O. Ciccolella, R. Anno et al., "Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets," *Nature Communications*, vol. 10, no. 1, pp. 1–12, 2019.

- [4] M. H. ur Rehman, C. S. Liew, A. Abbas, P. P. Jayaraman, T. Y. Wah, and S. U. Khan, "Big data reduction methods: a survey," *Data Science and Engineering*, vol. 1, no. 4, pp. 265–284, 2016.
- [5] P. H. Winne, "Leveraging big data to help each learner and accelerate learning science," *Teachers College Record*, vol. 119, no. 3, pp. 1–24, 2017.
- [6] C. Lehrer, A. Wieneke, J. Vom Brocke, R. Jung, and S. Seidel, "How big data analytics enables service innovation: materiality, affordance, and the individualization of service," *Journal of Management Information Systems*, vol. 35, no. 2, pp. 424–460, 2018.
- [7] S. Rajendran, R. Calvo-Palomino, M. Fuchs et al., "Electro-sense: open and big spectrum data," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 210–217, 2017.
- [8] W. Guo, J. Cai, and S. Wang, "Unsupervised discriminative feature representation via adversarial auto-encoder," *Applied Intelligence*, vol. 50, no. 4, pp. 1155–1171, 2020.
- [9] P. Y. Wu, C. W. Cheng, C. D. Kaddi et al., "Omic and electronic health record big data analytics for precision medicine," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 2, pp. 263–273, 2017.
- [10] L. Ulbricht, "When big data meet securitization. Algorithmic regulation with passenger name records," *European Journal for Security Research*, vol. 3, no. 2, pp. 139–161, 2018.
- [11] S. Maudsley, V. Devanarayan, B. Martin, and H. Geerts, "Intelligent and effective informatic deconvolution of "Big Data" and its future impact on the quantitative nature of neurodegenerative disease therapy," *Alzheimer's & Dementia*, vol. 14, no. 7, pp. 961–975, 2018.
- [12] Y. Zhang, M. G. Schaap, and Y. Zha, "A high-resolution global map of soil hydraulic properties produced by a hierarchical parameterization of a physically based water retention model," *Water Resources Research*, vol. 54, no. 12, pp. 9774–9790, 2018.
- [13] A. Mohamed, M. K. Najafabadi, Y. B. Wah, E. A. K. Zaman, and R. Maskat, "The state of the art and taxonomy of big data analytics: view from new big data framework," *Artificial Intelligence Review*, vol. 53, no. 2, pp. 989–1037, 2020.
- [14] X. Zhou, D. Liu, H. Bu et al., "XRD-based quantitative analysis of clay minerals using reference intensity ratios, mineral intensity factors, Rietveld, and full pattern summation methods: a critical review," *Solid Earth Sciences*, vol. 3, no. 1, pp. 16–29, 2018.
- [15] S. Wernisch and S. Pennathur, "Evaluation of coverage, retention patterns, and selectivity of seven liquid chromatographic methods for metabolomics," *Analytical and Bioanalytical Chemistry*, vol. 408, no. 22, pp. 6079–6091, 2016.
- [16] M. R. Bendre and V. R. Thool, "Analytics, challenges and applications in big data environment: a survey," *Journal of Management Analytics*, vol. 3, no. 3, pp. 206–239, 2016.
- [17] G. Dishon, "New data, old tensions: big data, personalized learning, and the challenges of progressive education," *Theory and Research in Education*, vol. 15, no. 3, pp. 272–289, 2017.
- [18] F. Stilo, E. Liberto, S. E. Reichenbach, Q. Tao, C. Bicchi, and C. Cordero, "Untargeted and targeted fingerprinting of extra virgin olive oil volatiles by comprehensive two-dimensional gas chromatography with mass spectrometry: challenges in long-term studies," *Journal of Agricultural and Food Chemistry*, vol. 67, no. 18, pp. 5289–5302, 2019.
- [19] X. Lu, D. Shankar, and D. K. Panda, "Scalable and distributed key-value store-based data management using RDMA-memcached," *IEEE Data Engineering Bulletin*, vol. 40, no. 1, pp. 50–61, 2017.
- [20] A. De Caigny, K. Coussement, and K. W. De Bock, "A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees," *European Journal of Operational Research*, vol. 269, no. 2, pp. 760–772, 2018.
- [21] M. Hatt, J. A. Lee, C. R. Schmidtlein et al., "Classification and evaluation strategies of auto-segmentation approaches for PET: report of AAPM task group No. 211," *Medical Physics*, vol. 44, no. 6, pp. e1–e42, 2017.
- [22] L. E. Kilpatrick and E. L. Kilpatrick, "Optimizing high-resolution mass spectrometry for the identification of low-abundance post-translational modifications of intact proteins," *Journal of Proteome Research*, vol. 16, no. 9, pp. 3255–3265, 2017.
- [23] R. Godwin-Jones, "Scaling up and zooming in: big data and personalization in language learning," *Language Learning & Technology*, vol. 21, no. 1, pp. 4–15, 2017.
- [24] K. O'Shea and B. B. Misra, "Software tools, databases and resources in metabolomics: updates from 2018 to 2019," *Metabolomics*, vol. 16, no. 3, pp. 1–23, 2020.
- [25] K. Uppal, D. I. Walker, K. Liu, S. Li, Y.-M. Go, and D. P. Jones, "Computational metabolomics: a framework for the million metabolome," *Chemical Research in Toxicology*, vol. 29, no. 12, pp. 1956–1975, 2016.